ISSUE 03 - JUL 2012

The MagPi™

A Magazine for Raspberry Pi Users

Fired up and ready to go

150+ GAMES TO TRY ON YOUR PI

- Debian Essentials
- The C Cave
- Scratch Patch
- Programming Fundamentals

ISSN 2051-9990

CC BY NC SA

The MagPi™

http://www.themagpi.com

Raspberry Pi is a trademark of The Raspberry Pi Foundation.
Front Cover and Feedback images were created using Photofunia

Welcome to Issue 3 of The MagPi, a community led magazine keeping you up to date with all things Raspberry Pi.

This month has seen the team working hard searching and testing 9000 stable Debian packages, and identifying over 150 games and 43 apps to try out on your Pi.

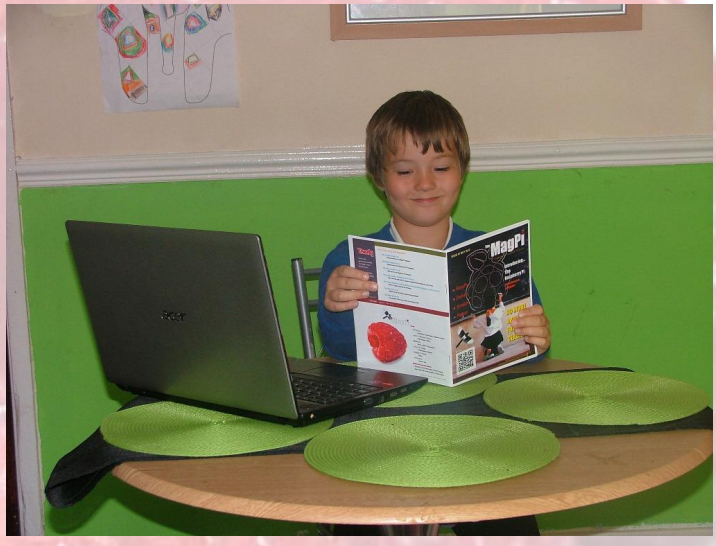
Meltwater introduces his article 'Meeting Pi', offering ideas and tips on introducing the Raspberry Pi to users of very young ages. We feature an article on low level programming using C and Alex Kerr has produced his article 'Programming fundamentals' describing how to use variables, 'If' statements and loops.

Bodge N Hackitt continues his series, explaining how to program the USB robotic arm using Python. Darren teaches us more on interfacing with the Pi, plus more from the reader's favourites - Command line, Scratch Patch and the Python Pit.

An email that caught my eye this month was from Cayton-John, aged 7, reading up on the first edition of the MagPi.

Could this be our youngest reader?

If you have photos reading our magazine, please send them to editor@themagpi.com.



On behalf of The MagPi team we want to thank Will Bengtson and his team at QRt for giving us our beautiful, personal The MagPi QR code. Please check out their site at http://QRt.co and see their clever generator and other attractive work.

Finally, keep your questions for Liz and Eben coming in and look for our official interview in next month's magazine.

We hope you continue to enjoy reading The MagPi and that it inspires you to get programming.

Ash Stone

Chief Editor of The MagPi

GETTING YOUR RASPBERRY PI TO AUTOBOOT TO LXDE WITHOUT LOGIN

If you are tired of logging in to your Debian SD card with username: pi, password: raspberry and then typing startx, you can get your Raspberry Pi to boot straight to the graphical interface without having to type anything...

After logging in type:
```
sudo nano /etc/inittab
```

Scroll down to the line:
```
1:2345:respawn:/sbin/getty 115200 tty1
```

Change it to:
```
#1:2345:respawn:/sbin/getty 115200 tty1
```
(Put a # at the beginning to comment it out)

Add a line under it:
```
1:2345:respawn:/bin/login -f pi tty1</dev/tty1>/dev/tty1 2>&1
```
Press CTRL+X and Y to save changes

Then type:
```
sudo nano ~/.bash_profile
```

Type in:
```
startx
```
Press CTRL+X and Y to save changes

Then
```
sudo reboot
```

Your Raspberry Pi should now boot up to LXDE automatically. I'm sure that will make many of you happy!

Make sure to follow these instructions carefully and be aware of the lack of security - the contents of your Raspberry Pi are now open to anyone that has access to it. More useful Debian tips are on the next page.

Article by Jaseman

Contents

Debian Essentials	P.04	The C Cave	P.22
Interfacing (In Control) Part 2	P.07	The Scratch Patch	P.24
Command Line Clinic	P.10	The Python Pit	P.26
Skutter Part 2	P.14	Programming Fundamentals	P.29
Protect Your GPIO Connector	P.16	Feedback	P.30
Meeting Pi	P.20	Weblinks & Credits	P.32

debian

ESSENTIALS FOR YOUR RASPBERRY PI

These procedures were tested with debian6-19-04-2012.img. Get the latest from http://raspberrypi.org/downloads

(You will need to have your Pi connected to the internet.)

REMOVING THE BLACK BORDER FROM AROUND THE SCREEN

This step is only required if you are finding a black border around the edge of your screen or the image overlaps the screen.

After booting up debian, login with
username: pi
password: raspberry

At the pi@raspberrypi:~$ prompt, type in:
sudo nano /boot/config.txt

Type the following into the nano text editor:
overscan_left=-10
overscan_right=-10
overscan_top=-10
overscan_bottom=-10

Press Ctrl+X to Exit and Y to say Yes to saving the changes.

When you are back at the command prompt type:
sudo reboot

Wait for the Pi to reboot and see if the borders have gone.

Repeat the process above changing the numbers by minus 10 each time until the border is gone. I found that my HDMI monitor required -40 for all settings, but my TV worked best with zero for left and right and -20 for both top and bottom. You will need to find the best settings to suit your particular screen. If the picture goes off the edge of the screen, just use positive numbers for the overscan values.

IF AFTER REBOOTING YOU ARE GETTING NO PICTURE:

You can reset the settings back to default by typing blindly:
pi
raspberry
sudo rm /boot/config.txt -R
sudo reboot

This will remove the config.txt file completely. After the reboot your picture should return. Try setting your overscan again with slightly lower numbers until you find the best values.

Alternatively, if you have a Windows PC and a card reader you can access the config.txt file and edit it with the Windows Notepad program.

You should also check the settings of your TV or monitor (4:3/16:9 aspect ratio, image adjust, pan and zoom settings).

INSTALLING THE SOUND DRIVER MODULE

If you intend to have sound through the TV, make sure the TV volume is up, otherwise connect a 3.5mm headphone cable from the Pi's analogue output to your speakers/stereo equipment and switch them on.

Type the following at the command prompt:
sudo apt-get update
sudo apt-get install alsa-utils (Answer 'Y' if asked about disk space used)
sudo modprobe snd_bcm2835

If you want sound through the HDMI cable type:
sudo amixer cset numid=3 2

Or if you want sound through the analogue (headphone) socket type:
sudo amixer cset numid=3 1

To make sure the sound driver module gets loaded each time you boot up type:
sudo nano /etc/modules

At the bottom of the file add the following:
snd_bcm2835

Press Ctrl+X and then Y to save the changes.

Reboot the Pi:
sudo reboot

Login and start the LXDE Graphical Interface by typing:
startx

When LXDE has loaded, select 'Music Player' (LXMusic) from the Sound & Video menu. Press the Play button. If all is well you should hear a 40 second piece of music called 'Cellule' by Silence.

You can use the File Manager to copy MP3 music from a USB memory stick into your home folder or play them directly through Music Player. Note, however, that it won't play WMA files.

FIXING GEANY (Python Editor)

When trying to execute Python scripts from Geany you will get an error.

To resolve this, go to Edit>Preferences>Tools.

Change the 'Terminal:' setting from 'xterm' to '/usr/bin/lxterminal'.

Click Apply and OK.

INSTALLING OMXPLAYER (Command Line Movie Player)

The OMXPlayer will allow you to play AVI and MP4 movies on your Raspberry Pi.

At the command prompt type:
sudo apt-get install omxplayer

Use File Manager to copy some movie files from a USB device into your home folder then from the command prompt type:
omxplayer filename

While playing press 'p' to pause and 'q' to quit. Use arrow keys to jump forward/back.

INSTALLING CHROMIUM WEB BROWSER

Chromium is slow on the Pi, but it works well with the following useful websites:
http://qwebirc.swiftirc.net - instant messenger chat rooms
http://www.dropbox.com - upload/download/share/transfer
http://www.gmail.com in Basic HTML mode - send/receive Google email with attachments

sudo apt-get install chromium-browser

INSTALLING XPDF

XPDF is a free PDF document viewer.

Download The MagPi PDF files by opening the Midori web browser and visiting http://www.themagpi.com. Click on one of the PDF download mirrors. Choose 'Save' and watch the progress of the download at the bottom right of the browser.

At the command line, type:
ls (check that The MagPi Issue X Final.pdf is listed)
sudo apt-get install xpdf (Type Y if asked about space)
xpdf "The Magpi Issue 1 Final.pdf" (Wait a moment and the PDF document should be displayed)

OTHER APPS AND GAMES

NOTE: Some of these get listed under 'Other' in the LXDE menu.

To install use: sudo apt-get install <name>
To uninstall: sudo apt-get remove <name>
To clean up: sudo apt-get autoremove

abiword Word Processor
amsn MSN Messenger (Slow to open/close but works)
audacity Audio Editor / Player
avifile-player Useless for video but plays WMA+MP3 music
brandy BBC BASIC V Programming (Type 'brandy')
bwbasic Bywater BASIC Programming (Type 'bwbasic')
espeak Command Line Text2Speech Synth
evolution Email Calendar Contacts Memos Tasks
extcalc Powerful Scientific Calculator
feh Command Line Image Viewer (With many options)
filezilla FTP file sharing client
fotoxx Photo Editor
fraqtive Mandelbrot Designer
galculator Calculator
gimp GNU Image Manipulation Program
gnumeric Spreadsheets
gpaint MS Paint-like drawing
grafx2 256-Color Paint Program
grisbi Personal Finance Management Program
homebank Manage Personal Accounts At Home
lifeograph Private digital diary
matchbox On-screen Keyboard
mc Midnight Commander - terminal file manager (Type 'mc')
milkytracker Music creation tool inspired by Fast Tracker 2
mtpaint Powerful Graphic Editor
openoffice.org Office Productivity Suite
oxine Media Center for MPEG, MP3 and WMA
qrencode QR Code encoder>PNG image (Type qrencode <string> -o filename)
schism ImpulseTracker Clone
scribus Desktop Publishing (Used to produce The MagPi)
ghostscript PostScript/PDF interpreter (Use with Scribus)
tuxpaint A Paint Program For Young Children
xball Simulate bouncing balls in a window (Click and drag in the window)
xchat IRC Chat Client
xcircuit Electrical Diagram Package
xine-ui Media Player for MPEG, MP3 and WMA
xpad Sticky Note Application
xpaint Simple Paint Program
xtrkcad Model Train Track CAD Program
yakuake A terminal window drops down from the top of the screen when you press F12.

3dchess Game (3x2d boards)
abe Abe's Amazing Adventure Game
ace-of-penguins Freecell, Solitaire+ Minesweeper, Teipei...
airstrike 2D Airplane Dogfighting Game
atom4 Color Puzzle Game
atris Tetris game
balazar3-2d Cool 3D Game
beneath-a-steel-sky Sci-fi Adventure Game (Nice Intro!)
blobwars Metal Blob Solid - Amazing 2D Platform Game
blockade Puzzle Game - command line 'sudo blockade'
bumprace 2D Space Maze Game
bygfoot Football Manager Game
ceferino Don Ceferino Hazaa Platform Game
childsplay Interactive Childrens Games Collection
crimson Crimson Fields Tactical Game
dangen Strange Shoot 'Em Up Game
dodgindiamond2 Arcade Shoot 'Em Up (Press M for Fire)
dossizola Isola Board Game
empire Text-Only Empire Building Game
enigma Puzzle Game
etw Eat The Whistle Football Arcade Game
fillets-ng Fish Fillets Puzzle Game
flobopuyo Connect4 meets Tetris
freealchemist Block Game
freedink RPG (Use GNU Free Dink from Other)
freedroid Paranoid Game Clone
frozen-bubble Frozen Bubble 2 Game
geki2 Xenon-like vertical shoot'em up (Fantastic!)
geki3 R-Type-like horizontal shoot'em up (Amazing)
ghextris A Tetrissudo-like Game On A Hexagonal Grid
glotski Slide Blocks To Reach A Goal
glpeces Tangram Puzzle Game Clone
gmchess Chinese Chess Game (Xiangqi)
gnugo The Game Of 'Go'
gnujump Plaftorm Game
gnuminishogi Mini Shogi 5x5 Board (Type 'help')
golly Game of Life Simulator
gravitywars Gravity Force Clone
grhino Othello/Reversi Board Game
groundhog Simple Logic Game
gtans Tangram Puzzle Game
gtkatlantic Game Like Monopoly
gtkballs Logic Game
gtkboard Many Board Games In One Program

Continued over page...

gtkpool Pool Billiard Game
hex-a-hop Hexagonal Tile Puzzle Game
hexalate Color Matching Puzzle
hexxagon Hexagonal Ataxx Clone
holotz-castle Mystery Platform Game
hrd HuaRongDao Puzzle Game
jester Board Game Similar To Othello
kball Game Of Skill And Reflexes
ketm 2D Scrolling Shooter
komi Komi The Space Frog Arcade Game
late A Bit Like Bally II
lbreakout2 A Fast Ball-and-paddle Game
liquidwar Multiplayer Wargame
lmarbles Build Figures Out Of Colored Marbles
lmemory A Children's "Memory" Card Game
luola Multiplayer Cave-Flying Game
madbomber A Kaboom! Clone
magicmaze Rescue Maiden / Avoid Monsters
magicor Puzzle Game (Slow)
mazeofgalious The Maze of Galious (Slow)
meritous Action-adventure Dungeon Crawl Game
micropolis Real-time City Management Simulator
mokomaze Ball-in-labyrith-game for the FreeRunner
moon-lander Game
mousetrap A Simple Game Of Ball Chasing
netmaze 3-D Multiplayer Combat Game
nikwi Platform Game (Good, But Slow Between Levels)
njam Pacman-like Game With Multiplayer Support
oneisenough 2D Platform Game - Epic Struggle Of Balls
open-invaders Space Invaders (Crashes Randomly)
openyahtzee Classic Dice Game Of Yahtzee
overgod Bi-directional Scrolling Arcade Game
pachi Platform Game Featuring Pachi el Marciano
pacman Chase Monsters In A Labyrinth
palapeli Jigsaw Puzzle Game (Slow To Load)
pangzero Pop Balloons With A Harpoon
pathogen Pathogen Warrior - Match 3D Model Structures
pathological Puzzle Game Involving Paths And Marbles
pegsolitaire An Education Game Similar To Hi-Q
penguin-command A Missile Command Clone
pente Five In A Row Game
petris Peter's Tetris - A Tetris(TM) Clone
pingus Free Lemmings(TM) Clone
pokerth Texas Hold'em Game
powermanga Vertical Shoot 'em Up With 3D GFX
quarry Board games Go, Amazons, and Reversi
rafkill vertical shoot'em-up
ri-li a toy train simulation game
sgt-puzzles Simon Tatham's Puzzle Collection
simutrans transportation simulator
slimevolley Unrealistic 2D Volleyball Simulation
snake4 Snake Game
solarwolf Collect The Boxes And Don't Become Mad
sopwith Port Of The 1980's Side-scrolling WWI Dogfighting
spacearyarya 3rd Person Shooter In Pseudo-3D
spout Tiny Abstract Black And White 2D Cave-shooter
starvoyager 2D 'Star Trek' themd arcade game
stax Collection Of Puzzle Games Similar To Tetris Attack
stroq A Polarium/Chokkan Hitofude Clone
supertux Classic 2D Jump 'n Run Sidescroller With Tux
tecnoballz Breaking Block Game Ported From The Amiga
teg Turn Bbased Strategy Game
tenmado Shoot 'em Up Game - Blue or Red World
tennix 2D Tennis Game
tictactoe tic-tac-toe Game Written In Ruby
tmw The Mana World Is A 2D MMORPG
tome Single-player, text-based Dungeon Simulation Game
toppler "Nebulus" 8/16-bit Clone
tumiki-fighters 2D Shooter (Very slow)
tuxpuck "Shufflepuck Cafe" Clone
tuxtype Educational Typing Game For Children
tworld Chip's Challenge Game Engine Emulation
typespeed Zap Words Flying Across The Screen By Typing
viruskiller Game About Viruses Invading Your Computer
vodovod Lead The Water To The Storage Tank
whichwayisup 2D Platform Game With A Rotational Twist
wing Galaga-like Arcade Game (CTRL=fire)
xasteroids X-based Asteroids-style Arcade Game
xblast Multiplayer Game (Only Works In Mini-mode)
xbomb A 'Minesweeper' Game With Shapes
xbubble A Puzzle Bubble Clone
xchain A Strategy Game For 2-4 players
xdemineur Puzzle Game
xevil Violent Side-scrolling Game
xinv3d 3D Space Invaders
xjig Jigsaw Puzzle (Middle mouse button to mirror)
xjump Jumping Game
xletters Type Falling Words Before They Land
xmahjongg Tile-based Solitaire Game
xmille The Classic Game Of Mille Bournes
xmpuzzles Collection Of Puzzles
xoids Asteroids Game (Right Shift To Fire)
xonix Bally II-like game
xpat2 Generic Patience Game
xscavenger A Lode-runner-like Platform Game
xshisen Shisen-sho puzzle game
xskat 3-player card game "Skat"
xsol X Solitaire
xsoldier shoot 'em up game (Amazing!)
xtron Tron game for X11
xvier a "Four in a row" game
xwelltris 3D Tetris like popular game similar to Welltris
zangband A single-player, text-based, roguelike game
zatacka Arcade multiplayer game like nibbles

We finished testing all the games, but ran out of time to test all the apps. We also had to remove the instructions for installing Quake III - the download site is no longer available. We are working on providing our own mirror of it. In the meantime, there should be plenty here to keep you entertained!

Special thanks to Chris 'tzj' Stagg for assisting me with the testing. I would not have got the list compiled without his help. Thanks also To Antiloquax - had he not sent me his spare Pi, I would not have been able to test anything.

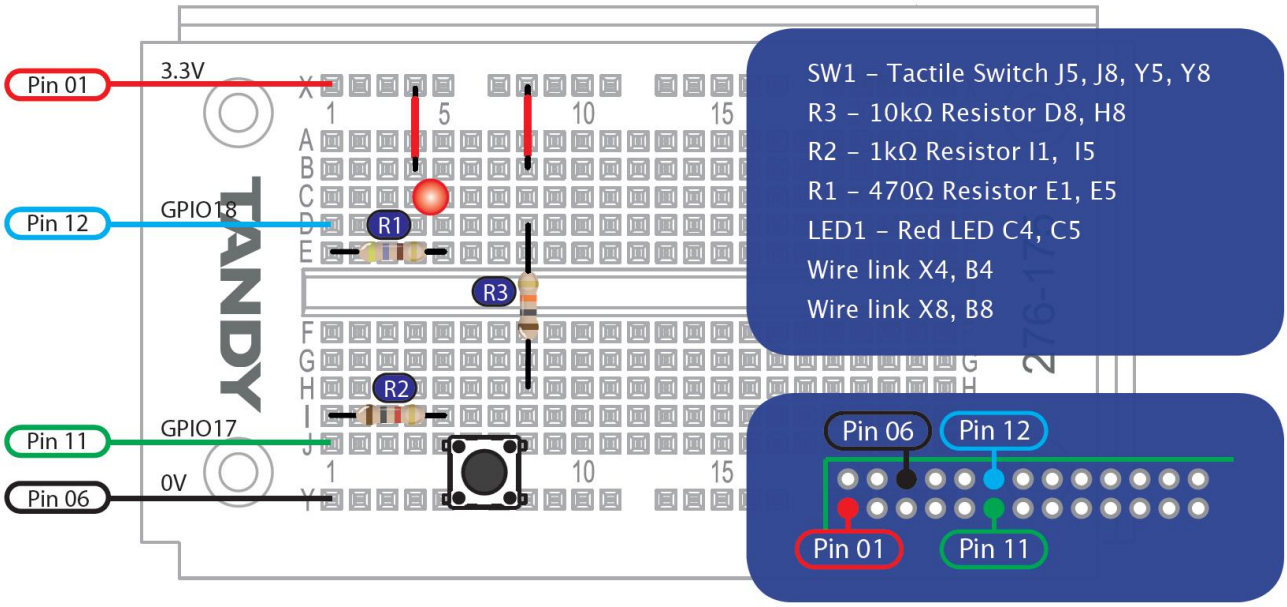
Article by Jaseman

IN CONTROL

INTERFACING PROJECTS FOR BEGINNERS PART 2

BY DARREN GRANT

In the first part of this series we looked at using the Raspberry Pi GPIO port as an input to monitor the status of a switch. In this part we learn about outputs.



We use the term INPUT to refer to the fact that the computer is being controlled in some way by an external event, such as our previous experiment where the computer responds to the press of a switch. An OUTPUT is where the computer is controlling or communicating with the outside world.

You are perhaps used to thinking of things like the screen and keyboard as being part of a computer but they are in fact peripherals, extra parts that make it possible for the computer to communicate with the outside world. A computer is just a collection of chips on a circuit board that has very little use until it has a way of communicating with the outside world. A computer that can do nothing more than talk to itself is of little use. Put some software on an SD card and plug the power in and the Raspberry Pi is a complete operating computer. However, without a screen we have no way of knowing what it is doing. A screen is an output device - an example of just one of the many ways that a computer can provide an output.

As we learnt last time, computers are binary devices so a computer communicates with the outside world by switching things on and off. The image you see on the screen is produced by the computer switching the individual dots (pixels) on and off on the screen. We are going to have the Raspberry Pi control the LED on our breadboard, from last month's experiment.

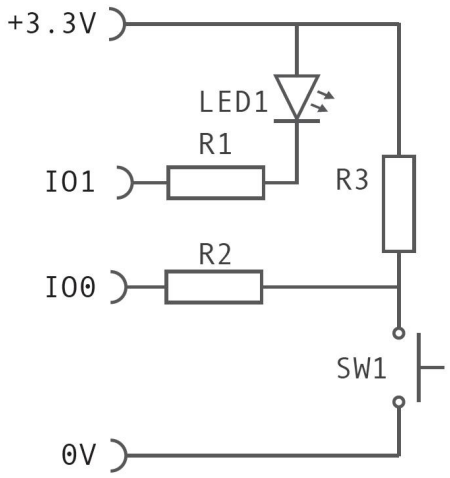
Circuit Description

We are making two small changes to the circuit from last month. We turn resistor R1 90° so it is connected to points E1 & E5. This disconnects the LED from the switch. Next, we connect a second GPIO pin to point D1 so that the LED can be independently controlled.

When using the Raspberry Pi GPIO pins as an output we have to be very careful not to draw too much current. The 470Ω resistor in this circuit limits the current to 3mA, just enough to light a standard 3mm Red LED.

Continued over page...

IMPORTANT
Before connecting anything to the Raspberry Pi please be aware that incorrect connections could cause damage. Please take care.



Python LED Test Program

Create a new text file named ledtest.py and enter the following program.

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

GPIO.output(12, False)
time.sleep(3)
GPIO.output(12, True)
```

Program Description

We start by including the time and RPi.GPIO packages in our program so we can make use of their functions by using import. Next we configure GPIO pin 12 on the Raspberry Pi as an output.

Our program is a simple test; it sets pin 12 LOW then waits 3 seconds before setting it HIGH. This will switch the LED on for 3 seconds and then off when the program ends. It might seem a little odd that the LED lights up when we set our output as FALSE. This is because one side of our LED is connected to the 3.3V power. To complete the circuit our GPIO pin has to be 0V. When we set the value as TRUE our output is set to 3.3V making both sides of the LED 3.3V preventing a current flowing.

To start the program type the following command into the terminal window.

```
sudo python ledtest.py
```

The sudo command is necessary as programs that use the GPIO port require superuser privileges within the operating system.

Pushbutton Test Program

Now we can combine the input and outputs to reconnect the LED to the switch in software. Create a new file called pushbutton.py and enter the following program.

```
#!/usr/bin/python
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)
GPIO.setup(12, GPIO.OUT)

while True:
    if GPIO.input(11):
        GPIO.output(12, True)
    else:
        GPIO.output(12, False)
```

Program Description

We start by importing the RPi.GPIO package and configuring GPIO pin 11 as an input and pin 12 as an output using the now familiar “GPIO.setup” syntax.

By using "while True" we create a never ending loop so that everything below this will be repeated until we choose to stop it.

Our program keeps checking the status of pin 11. This will always be True (High) while the button is not being pressed. As soon as the button is pressed the GPIO pin goes low and our result will be False. At this point we set GPIO pin 12 to be false (Low) to light up the LED.

When you have had enough press the CTRL+C keys to stop the program.

Having the LED light up when the switch is pressed is not terribly useful as it just gives us the same result that we had before when the LED was directly connected to the switch. Let's make something a bit more interesting.

Electronic Die

Because the LED and switch are now independent, we have a great deal more flexibility. We will now write a program that will flash a random number between 1 and 6 when the button is pressed.

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import random
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN)
GPIO.setup(12, GPIO.OUT)

while True:
    if not GPIO.input(11):
        flash = random.randint(1,6)
        while not GPIO.input(11):
            GPIO.output(12, True)
        while flash > 0:
            GPIO.output(12, False)
            time.sleep(.5)
            GPIO.output(12, True)
            time.sleep(.5)
            flash -= 1
    else:
        GPIO.output(12,True)
```

Program Description

The program runs a continuous while loop waiting for the button to be pressed. Once the button press is detected we generate a random integer value between 1 and 6. An integer means a whole number rather than a floating point number, so we get numbers like 2 and 3 rather than 2.45678. Once the random number has been generated, we wait for the button to be released before continuing, to prevent cheating.

When the button is released we enter a countdown loop that flashes the LED and reduces the count by 1 after each flash, until it reaches 0. The program then returns to the main loop where it waits for the next button press to start the process again.

Conclusion

We have now covered the basics of how to monitor an input and switch things on and off using an output. With just this simple set-up of one switch and one LED we can already make something useful.

Why not try modifying the pushbutton.py program so that the LED is on but goes off when the button is pressed. There is more than one way of doing it. Can you see two or more possibilities?

Many people have asked about how to control something that requires more power than a small LED. In the next instalment we will look at the various options available. ●

Tandy®
www.tandyonline.co.uk

This article is sponsored by Tandy.
All the components mentioned in this series and much more are available from the new Tandy online shop. Many new components being added daily.

Command Line Clinic

By Bobby Redmond (bredman)

In this issue, we will explain why Linux commands behave the way they do. We will also explore how to join commands together as building blocks to do some useful things.

The Linux command line is not very polite or friendly. It never tells you if things went well and spits out gibberish when things go wrong. This month's article tries to explain why the Linux command line behaves this way and will help you to understand why it looks the way that it does.

The Linux command line is so unfriendly you probably wondered why nobody ever improved it to make it more human-friendly. The reason is that the output from the command line was never designed to be read by a human - it was designed to be read by computer programs. Let's take an example of a really unfriendly command and try to make sense of it.

One of the most useful commands in Linux is the find command. It allows you to find a file if you know the name or part of the name. As an example we will try to find every C program in the system. A C program should have .c at the end of its name. So let's try this command…

```
find / -name *.c
```

This should find all files named "*.c" (where * means we will accept anything) starting from the top of the file system (/). Unfortunately, all you get is a heap of meaningless rubbish. This is the sort of thing that gives the Linux command line such a bad name. How can a command be so useful if you can't read the output?

First of all, we have to get rid of all the errors which are shown. You may notice that all of the errors include the text "Permission denied". This means that the user does not have the authority to do something. When you see "Permission denied" it is often a good idea to try adding the word sudo to the beginning of the command. This tells the command line that you want to run the command as a superuser, not a normal user. So let's try this command…

```
sudo find / -name *.c
```

At last the errors are gone and we have a list of all of the C files on the computer. But it's not very friendly. You just get a list, nothing at the beginning to say "Here is a list of your C programs" and nothing at the end to say "End of list, have a nice day". This list was obviously never intended to be human-friendly. Why was it designed this way? If not intended for a human to read, who was it intended for?

One of the magical things about Linux is that the output from one command is designed to be used as the input for another command. This allows us to join lots of simple commands together to create more complex results. As an example, let's take the list of C programs and send the list to a file. Try this command…

```
sudo find / -name *.c > mylist
```

How to enter very long commands

As a beginner, you may never expect that you will be writing very long commands. But in this article you can see that simple commands can be joined together to create very long commands.

Some commands may be so long that they are wider than your screen. In this case, just keep typing and let the text flow to the next line. Do not press the Enter key until you have finished typing the command.

You should see no response. What the “>” symbol means is that the result should be sent to a file called “mylist” instead of your screen. We can see what is stored in the file by using the command…

```
cat mylist
```

You should see that this file now contains the list that should have been sent to your screen. So why is this useful? We can now run another command on this file, for example to sort the list. Enter the following command…

```
sort mylist
```

To find if the word “hello” appears in the list, we could use the command…

```
grep “hello” mylist
```

To count how many lines are in the list, we could use the command…

```
wc -l mylist
```

This shows that it is possible to run one command which works on the output of another command. However it is very inconvenient to need to create a temporary file every time. Luckily, there is a way to join the output of one command to the input of another command. It is known as a “pipe” and is represented by the symbol “|” (see the top of page 13). Let’s see some examples.

A sorted list of all the C programs...

```
sudo find / -name *.c | sort
```

A count of all the C programs...

```
sudo find / -name *.c | wc -l
```

How many C programs contain the word “hello” in the filename?

```
sudo find / -name *.c | grep “hello”
| wc -l
```

This ability to join commands together is one of the most important aspects of the Linux command line. Just a few commands can be joined together to create an infinite number of useful solutions. It is just like giving bricks, timber and glass to a builder; there is no limit to what can be created when the pieces are joined together.

This also answers another question; if you ask two Linux experts how to solve a problem, why do you get two (or more) answers? It is because the basic building blocks can be joined together in so many ways to answer your question.

The most popular way to join commands together is by using pipes, but let's explore some other methods.

To create a sequence of commands, the symbol “;” may be placed between the commands. To illustrate this, let’s create a nice human-friendly list of C programs...

```
echo “Here is a list of your C
programs” ; sudo find / -name *.c ;
echo “End of list, have a nice day”
```

You can even place brackets “( )” around a sequence of commands to treat them like a single command. For example, to send the result of a sequence of commands to a file…

```
(echo “Here is a list of your C
programs” ; sudo find / -name *.c ;
echo “End of list, have a nice day”)
> mylist
```

Continued over page...

How to edit commands

You will quickly notice that you are typing the same commands again and again. This may be because you are making small changes to the command or you may be trying to fix errors.

It can be very frustrating to type the same long command several times. Luckily, there is an easier way. Press the up arrow to scroll backwards through a history of your recent commands. You can move around in a command using the left and right arrow keys to change text.

A more complex method is to use the output of one command as a parameter to another command. This can take some getting used to. Let's start with a very simple example.

To show how many files are in your current directory, use the command...

```
ls | wc -l
```

To print this on the screen, use the command...

```
echo `ls | wc -l`
```

This uses the ` symbol (see the top of page 13) to use the output of the ls command as a parameter for the echo command. It does not look any different, but we could build a more complex command...

```
echo "There are `ls | wc -l` files in this directory"
```

To show the usefulness of this method, here is a command to search for any C program which contains the word "buffer". This can be very useful if you want to find which file uses a particular variable or function...

```
grep -l "buffer" `sudo find / -name *.c`
```

Sometimes, the result of a command can be too large to fit on one screen. As an example, let's print a list of all the Python (*.py) programs in the system…

```
sudo find / -name *.py
```

You will see that there are a lot of Python programs and you have no chance of reading them all on one screen. Luckily, there is a way to stop the output to give you a chance to read the text. You can pipe the result to the command "more"...

How to deal with errors

In the above text, we dealt with the errors in the find command by using the sudo command. To be correct, this is cheating. It is a pure coincidence that sudo prevents the errors for the find command. This solution will not work for every other command that you try.

To know how to hide errors, you need to know a bit about how the Linux commands work. Every Linux command has input, output and errors. Normally the output and errors are mixed together, but they can be separated if you wish. Earlier we saw that the ">" symbol sends the result to a file. If you need more control, you can use the "1>" symbol to send the output to one file, and the "2>" symbol to send the errors to another file. As an example, try the following commands…

```
find / -name *.c 1> mylist 2> myerrors
```

```
cat mylist
```

```
cat myerrors
```

As you can see, this sends all of the errors to the file named myerrors. This is slightly inconvenient because we must delete this file later (the command is "rm myerrors"). Luckily, there is a file designed especially for unwanted output called /dev/null. This is basically a bottomless pit where you can dump anything you don't like.

So, by using "2> /dev/null" we can be sure that the errors will be dumped and we don't need the sudo command any more. The example above, to count all of the C programs, would look like...

```
find / -name *.c 2> /dev/null | wc -l
```

```
sudo find / -name *.py | more
```

The command “more” lets you read through the result at your own pace. Press the space bar for the next page, or the Enter key for the next line. Press “q” to quit.

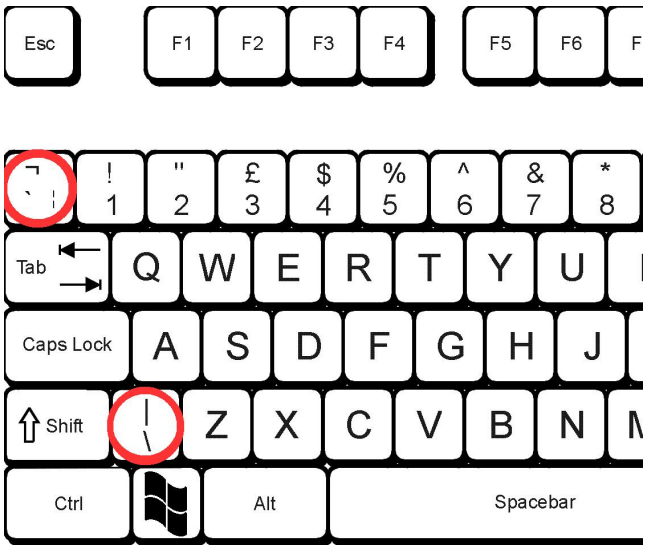
The command “more” is very limited, most people use its more modern replacement named “less”. The command is very similar to use…

```
sudo find / -name *.py | less
```

The “less” command has the advantage that you can use the arrow up/down keys and the page up/down keys. You can press the keys “<” (jump to start of list), “>” (jump to end of list) and “q” (quit).

Where are the ` and | keys?

If you have trouble finding the ` and | keys, they are circled in red in the diagram below for a UK keyboard. These are keys that you would normally not use if you are not familiar with the Linux command line.



Some useful commands

Here is a reminder of the commands that were used in this article. To learn more about a command you can read a reference manual by typing man followed by the command. For example, type "man find" to learn about the find command. Press the "q" key to exit the manual.

`cat file`	Show the contents of a file
`echo "text"`	Print text on the screen
`find directory -name x`	Find all files named x under a directory
`grep text file`	Find text within a file
`less file`	Show the contents of a file, allow the user to move up and down through the file
`ls directory`	Show a list of files in a directory
`more file`	Show the contents of a file, allow the user to move down slowly through the file
`rm file`	Remove (delete) a file
`sort file`	Sort the contents of a file
`sudo command`	Run a command as superuser
`wc file`	Count number of lines, words and characters in a file

SKUTTER: Part 2

How to write a program for your USB device

By Bodge N Hackitt

Installing Python PyUSB libraries

To be able to control your USB device you will need the Python PyUSB libraries. The PyUSB libraries have been written in collaboration with a team of 'open-source' volunteers.

To download them, open the Midori web browser and visit: http://goo.gl/5GBcD This will give you the option to save the pyusb-1.0.0-a2.tar.gz into the Downloads folder.
Type 'cd Downloads'

Then expand the compressed tarball file:
tar xvf pyusb-1.0.0-a2.tar.gz

List the files and directories by typing 'ls'. You should see a new directory called “pyusb-1.0.0-a2”.

Change directory by typing: 'cd pyusb-1.0.0-a2'
Type 'ls' again to see the contents of the directory.
Type 'sudo python setup.py install'

You can now start writing the program to control your USB device.

Writing the Python program

Type this code into Idle, the Python editor, and save it as 'arm.py':

```
# ROBOT ARM CONTROL PROGRAM
# import the USB and Time libraries into Python
import usb.core, usb.util, time
# Allocate the name 'RoboArm' to the USB device
RoboArm = usb.core.find(idVendor=0x1267,idProduct
=0x0000)

# Check if the arm is detected and warn if not
if RoboArm is None:
    raise ValueError("Arm not found")

# Create a variable for duration
Duration=1

# Define a procedure to execute each movement
def MoveArm(Duration, ArmCmd):
    # Start the movement
      RoboArm.ctrl_transfer(0x40,6,0x100,0,ArmCmd,
1000)
    # Stop movement after waiting specified time
    time.sleep(Duration)
    ArmCmd=[0,0,0]
      RoboArm.ctrl_transfer(0x40,6,0x100,0,ArmCmd,
1000)
```

```
# Give the arm some commands
MoveArm(1,[0,1,0]) # Rotate Base Anti-clockwise
MoveArm(1,[0,2,0]) # Rotate Base Clockwise
MoveArm(1,[64,0,0]) # Shoulder Up
MoveArm(1,[128,0,0]) # Shoulder Down
MoveArm(1,[16,0,0]) # Elbow Up
MoveArm(1,[32,0,0]) # Elbow Down
MoveArm(1,[4,0,0]) # Wrist Up
MoveArm(1,[8,0,0]) # Wrist Down
MoveArm(1,[2,0,0]) # Grip Open
MoveArm(1,[1,0,0]) # Grip Close
MoveArm(1,[0,0,1]) # Light On
MoveArm(1,[0,0,0]) # Light Off
```

To try out this code type into the command line:
'sudo python arm.py'
sudo is required as USB requires root access.
Give the arm room to move BEFORE testing.

You will notice that the ArmCmd has 3 values:
1. controls the motors for the grip, wrist, elbow and shoulder.
2. is for rotating the base. Valid values are 0 (No rotation), 1 (Anti-clockwise) and 2 (Clockwise).
3. is for the LED at the end of the arm. Valid values are 0 (Off) and 1 (On).

For the first value, you can combine movements by adding the commands together, for example [32,0,0] + [8,0,0] = [40,0,0]. This would make the elbow and wrist go down at the same time. Of course some combinations won't work, like attempting to operate the same motor in both directions.

For example, try MoveArm(1,[144,1,0]) and MoveArm(1,[96,2,0])

The idVendor and idProduct values (0x1267) & (0x0000) were obtained by typing at the command line:
'sudo lsusb -v'
This provides a list of details about all of the USB devices that are connected to your Raspberry Pi.

The values for '.ctrl_transfer' (0x40, 6, 0x100, 0) were obtained by USB sniffing the device using software such as 'Snoopypro' for Windows. These hexadecimal numbers represent the USB protocol for the robot arm. You can think of it as the language that the arm's circuit board controller understands.

The value 1000 at the end of .ctrl_transfer is a timeout value for sending the command to the USB device.

This is the bare minimum of what you need to be able to do to start programming your robot arm. If you wish to take your knowledge further, and eventually have the ability to accomplish much more, I have included some

extra information about how USB devices such as our robot arm actually work, to help you get started off.

An introduction to USB devices

All of the robot arm's motors are connected by wires to the circuit board. By sending signals to the circuit board we can control each of the motors. This circuit board is a USB device.

The protocols and methods used by USB are very complicated and events happen so quickly that it takes some clever software such as a USB sniffer to allow us to monitor what is going on. This port is also so strange that no one can even agree on who invented it or why! Some say it was designed to cut down the number of wires poking out of the back of your computer and it looks like the “Apple Desktop Bus” from the early Apple computers – so it must have been invented by Apple. Others disagree and point out that it works like a 10 Base T ethernet (computer network) with a star topology so it can’t be anything to do with Apple. I don’t think this argument will ever be won.

USB devices have a “Vendor ID” and a “Product ID”. These ID's have to be issued by a special organisation (USB.org) and they differentiate between all the different devices on the market and allow your computer to recognise what it is when you plug it in. That's why when you plug a mouse into your computer it knows that it's an “Acme Optical Mouse” and not an “Eastern Electric My Personal Storage Pad”, or in our case an “OWI Edge Robot Arm”.

Within the circuitry for USB devices are "endpoint addresses". USB devices use “pipes” which are dedicated lines for transmitting data to and from the device and computer. At the end of the pipe in your computer is a USB “hub” which works in a similar way to a network hub. It allows devices to be attached to branches that all trace back to one common point. The "endpoint" is at the other end of the pipe in our USB robot arm device and the address specifies where it is. Some USB devices are called “composite” devices because they combine two or more different devices together (like a web-cam which also includes a built in microphone). Such a device would have at least two end point addresses, one for the web-cam part and one for the microphone part.

When we are working with composite USB devices we may also need to refer to an interface number. Where a USB device is really made of several devices working together, and each device is at an endpoint, an interface groups those endpoints together to allow the separate devices to work together to perform a single function. The interface is referenced by an “interface number”. This number is the index for the device so that the USB controller knows which interface to talk to. The first interface number is “0”. The second interface is at “1” and so on.

As far as the control computer and USB is concerned, our robot arm is a single device (it is a circuit board which switches some electric motors on and off when it receives the right signal from the USB) and so it is very simple to write USB control programs for it.

You can consider that the circuit board included with the robot arm is like three rows of eight switches (or otherwise known as three bytes). When the computer sends the right signal to it via the USB it turns one or more of these switches on and in turn this starts a motor on the robot arm turning in one direction or another.

The robot arm circuit board actually uses a numbering system called "binary". When you send a control code such as [128,0,0] these numbers are converted into their binary equivalent. For example, this particular command would become [10000000, 00000000, 00000000]. The "1" in the first binary number is equivalent to turning on the switch that would start the shoulder motor turning.

When data is sent to and from any USB device the information is always sent in a series of bytes, or eight "on" and "off" / "1" or "0" signals like this. These signals are interpreted by USB devices in different ways and allow them to perform all of the different kinds of functions that they are capable of.

Next steps for the Skutter

At the moment, even with our own Python programs controlling our robot arm, it is still not much more than a toy. It still has no sense of where it is or what it is doing. Eventually we want our robot to be able to accomplish “missions”.

Our robot is going to have a number of sensors added to it to monitor position as well as to include a kind of robotic sense of sight. The values from these sensors will need to feed back into its Raspberry Pi “brain” via the GPIO. If you have started following the "In Control" series in this magazine you may be able to imagine how we could start to accomplish this.

Our Skutter Master Control Program will need to take these sensor values, interpret them and make decisions about what to do before directing the arm and motorised base about where and what to move and how much to move by.

In the next article we will investigate a number of ways in which we can provide this required "sensory feedback" to allow the robot arm to become much more than a simple toy.

In future articles we will look at ways of adding a motorised base with wheels to the robot arm and finally we will examine some ways of giving our "skutter" some kind of artificial intelligence. I will be writing these articles as I continue to build up my own version of the robot. I hope they will be of use to you or encourage you to start your own robotics project. Looking forward to seeing you all again soon!

To Protect and Serve Your Raspberry Pi

In this article I will describe a cheap way to protect your GPIO connector from accidental shorts and physical damage, and at the same time provide a GPIO socket which can be easily connected, at almost no cost.

What You Are Going To Need

Components

• 2 x 26 way IDC (insulation-displacement connector) ribbon socket.

• A length of ribbon cable (with 24 wires or more). An old floppy drive cable, or an old IDE hard-drive cable is ideal (older type with 40 wires, rather than the finer 80 wire ones).

• (Optional) A few inches of 2mm heatshrink sleeving, or failing that, insulating tape.

• (Optional) An IDC pin header, I rescued mine from an old motherboard.

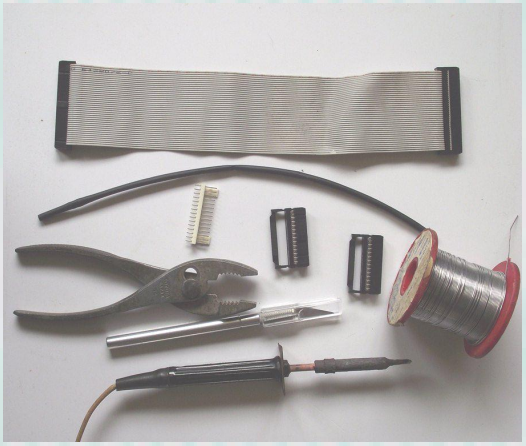
Tools

• A sharp modelling knife.

• A workbench vice or, if very careful, a pair of pliers / molegrips.

• (Optional) A few inches of solder.

• (Optional) A basic electronics soldering iron.

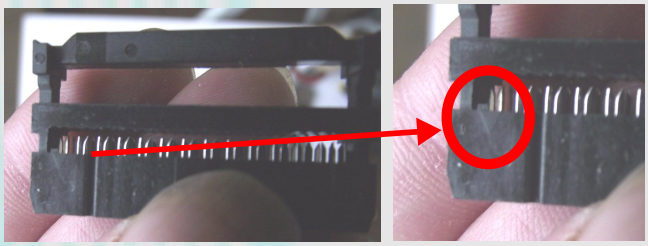


Getting Started - Preparing Your Cables and Connectors

1. Take your ribbon cable and locate the marked wire indicating wire number 1.

2. From the marker wire count off 26 wires in the ribbon then mark the gap between wires 26 and 27.

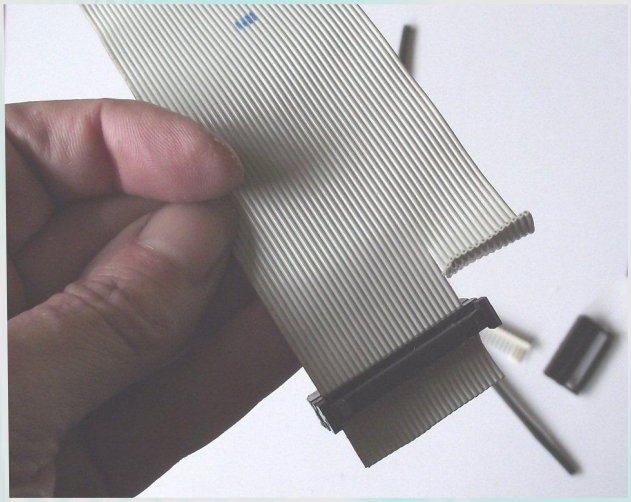
3. Take an IDC connector and identify pin 1 (look at the image at the top of the next column; under the left-most pin you will see a small triangle - thats pin 1).

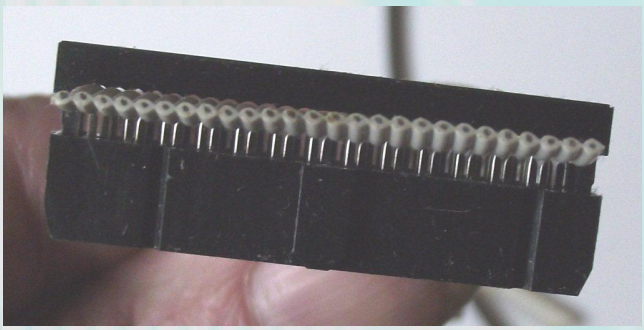


4. If your cable has connectors already on it, carefully cut them neatly off (either with a knife or pair of scissors).

5. Offer up the marked ribbon and ensure that wire 1 lines up with the connector pin 1 and that pin 26 is where you have marked it to be. Double check before cutting carefully between wire 26 and 27 for about 25mm (I find it easier to press the knife into the cable at the 25mm point and cutting back to the end).

6. Take the two ends of the ribbon in each hand and pull the two pieces apart (in a tearing action) slowly and carefully, producing a 26 way ribbon and a remaining ribbon.



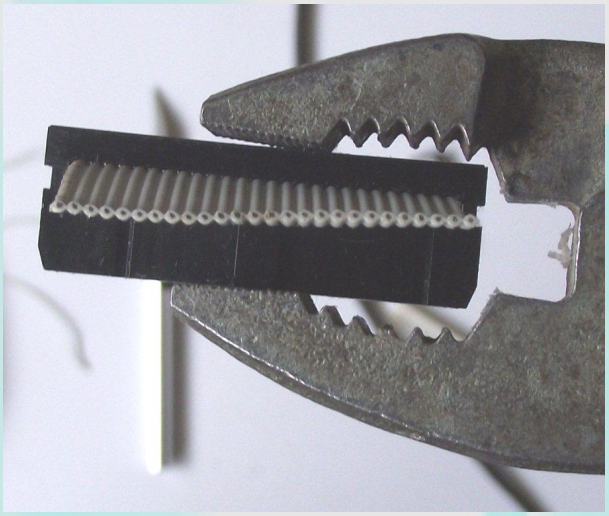


7. Insert the 26 way ribbon into an IDC connector, ensuring the marked wire and the marked pin (wire 1 and pin 1) are lined up.

Leave a little bit of cable protruding through the connector.

The Tricky Bit - Making Connections

I usually use a small bench vice for this, but failing that a pair of pliers or molegrips will suffice.



1. Make sure the ribbon is square with the connector.
2. If using pliers, with a piece of card protecting the connector (not shown in the picture):

Gently squeeze one end of the connector until the cable starts to push the wire between the spikes. Now go to the other end of the connector and repeat. Do the same about a third and two thirds along the connector. Continue this until the side clips latch into position. Ensure the wires are evenly squashed between the body and clamp of the connector. Remember it is essential to keep an even progress across the width of the connector.

If using a vice, close the vice until it is just holding the cable between the connector body and the clamp. Ensure for the last time that pin 1 and wire 1 are where they should be and that the cable is square with the connector. Tighten the vice until the connector is latched.

3. Determine how long you want the cable to be. For good signals don't make it too long - less than 15cm (6 inches) is ideal. Position the second IDC connector where you want it, ensuring it is the correct way round and repeat the clamping process.

4. Now carefully trim the cable ends as close to the connector as you can with the modelling knife.
5. If you have a multimeter available it is advisable to check through the connections to be 100% sure there are no shorts or bad connections.

GPIO Connector for your Raspberry Pi? ...Done!

Gert van Loo has posted a YouTube video showing making a GPIO cable using a vice, available here: http://goo.gl/DC4Sf

Wires - Easy Breadboard and Header Links

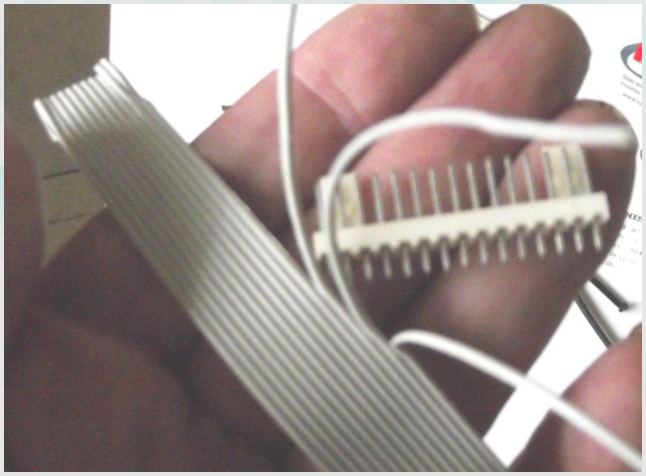
Now you have a neat 26 way socket in which you can easily connect wires to and from breadboards and other circuits. The simplest option is to buy some single solid core insulated wire (0.6mm/22 AWG is ideal). Single core is recommended as wires with multistrands are difficult to insert into the socket holes.

You can buy large rolls of wire from electrical suppliers or there are several eBay sellers which will provide a range of colours and lengths by the meter.

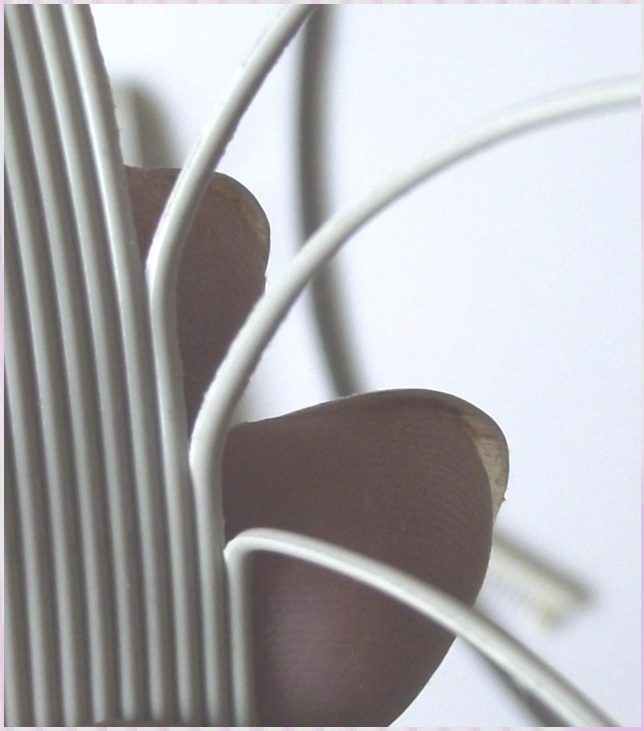
Alternatively, you can follow the instructions below to create your own flexible pin connectors, which makes great soldering practice too.

Optional Extra - Making Pin & Wire Connectors

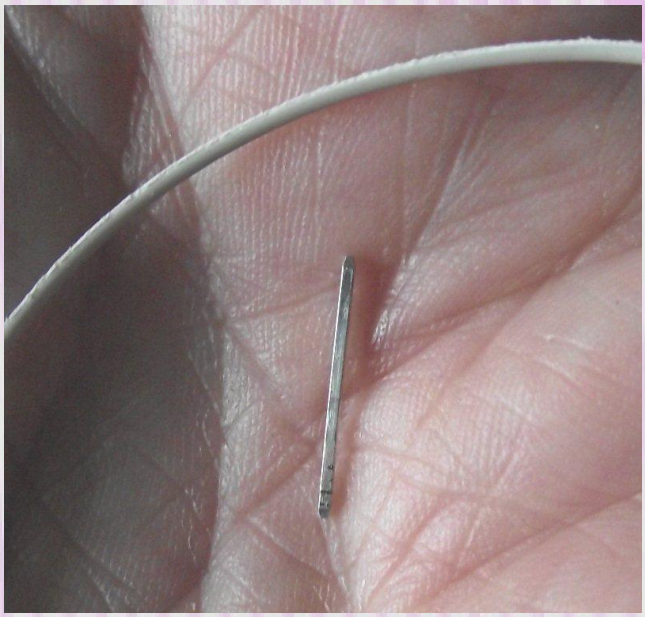
1. First take the IDC header and carefully pull the individual pins from the header using your pilers. Straighten any which get bent with the pilers.
2. Next we will take the remaining ribbon cable and strip it into individual wires. With the modelling knife, do the same as we did previously to split out the ends.



Continued over page...

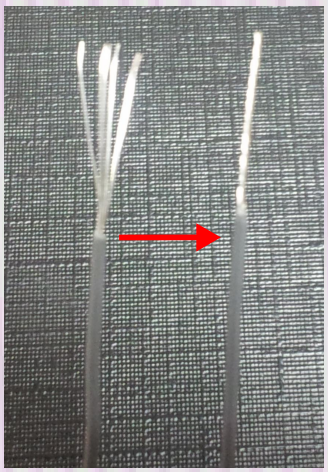


3. Carefully separate out each individual wire, being careful not to rip or break the insulation.

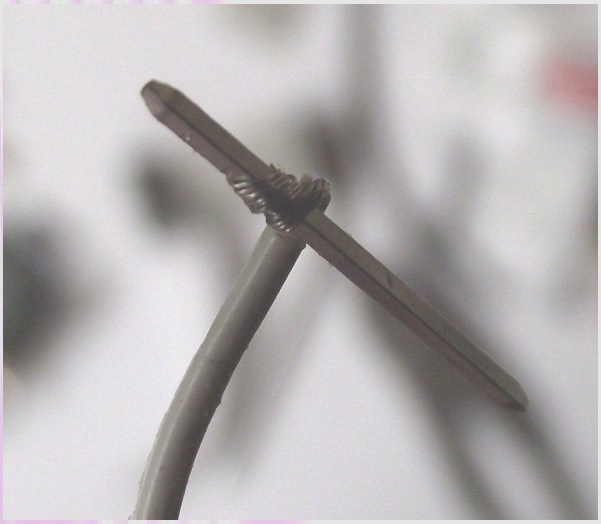


4. Take a length of the seperated wire and carefully remove about 15mm of insulation from each end.

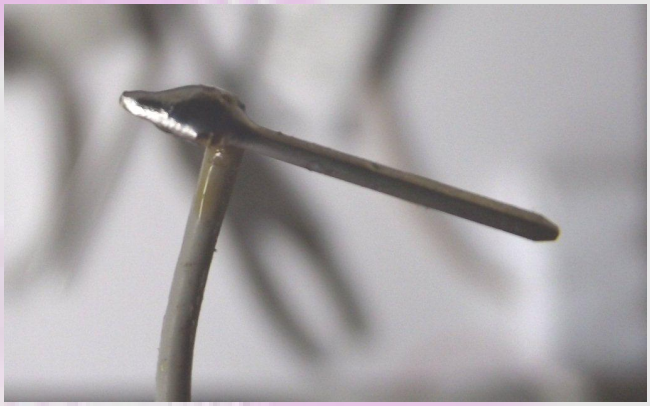
5. Twist the strands together on each of the ends.



6. Take a pin, put the twisted end of the wire in the centre of the pin and twist the bare wire around the pin toward the end.

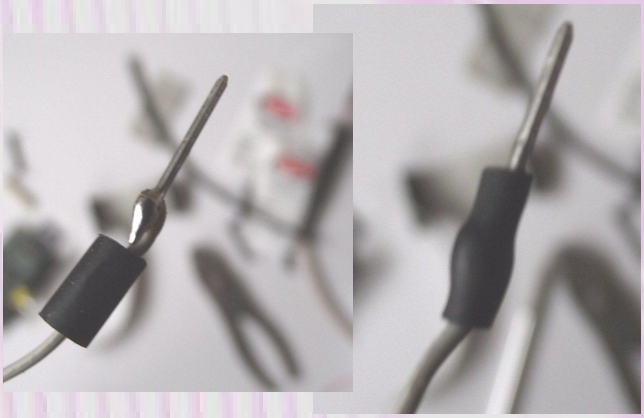


7. Taking care not to burn yourself (the pin will get very hot!), solder the wire to the pin. Ensure you trim any stray strands, if you have any. Do the same for the other end.



8. Take the heatshrink sleeving and cut two lengths about 5-10mm long.

9. Now slide a length of heatshrink sleeve over the soldered joint leaving about 10mm clear to the end of the pin (the wire should come out one side and the pin the other side). Next apply a little heat to shrink the sleeve over the joint. The side of the soldering iron's tip will suffice, a heat gun or even a hair dryer will work.



Your connection wires are now complete!

Repeat for as many wires as you want to make.

Important Piece Of Pi - GPIO Socket

The pinout of the GPIO Socket is now subtly different to the board's pin header.

WHILE THE PIN NUMBERS ARE STILL CORRECT THE VIEWED POSITION IS NOT.

Pin 1 is still Pin 1 but its on the opposite side to the Raspberry Pi pin header (as shown on the RPi Wiki, http://goo.gl/E19X9). To save yourself some heartache, it is recommended you clearly label Pin 1 and you keep a copy of the following pinout diagram handy.

Here is the corrected pinout for the GPIO SOCKET viewed into the socket. Note, pin 1 is the 3v3 pin and the connector's "key" is shown (by GPIO22) which also indicates which side Pin 1 is on.

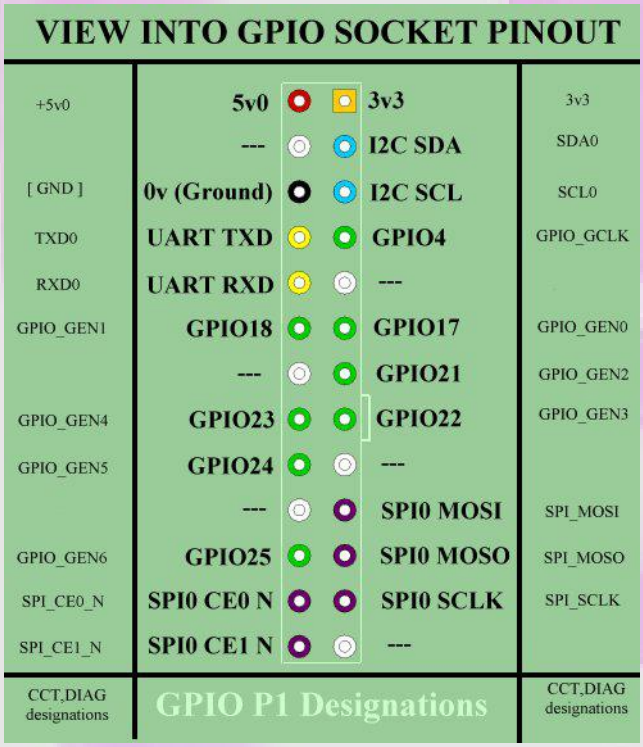
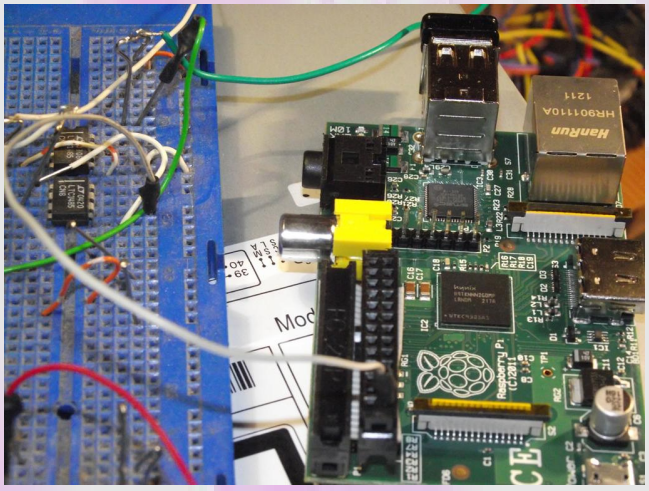
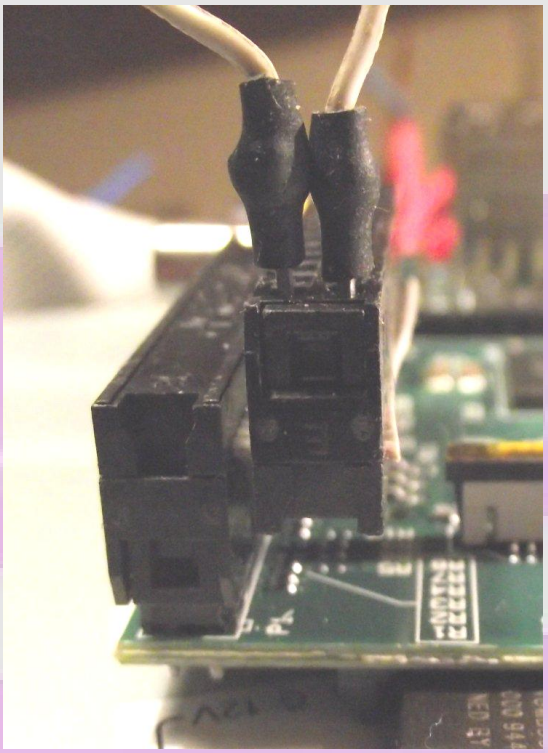


Image above: GPIO Socket Pinout

Image below and top right: A short GPIO header in use.

Image bottom right: A Pi-to-Pi GPIO connection (Don't do this!)



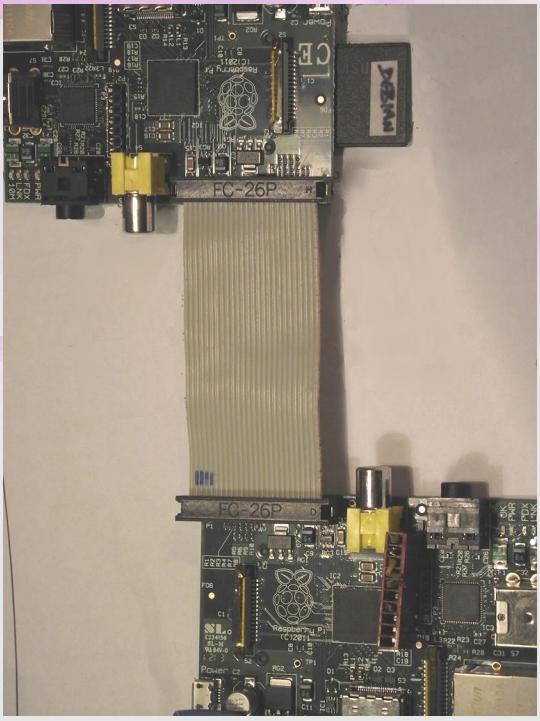


Final Warings - Double Check

Satisfy yourself that all is well before connecting ANYTHING to the socket. Check your connections and if possible test any voltages (always between 0v and 3v3) and/or currents present on the circuit.

These GPIO pins connect directly to the Raspberry Pi processor with very little electrical protection! The "magic smoke", once let out, cannot be replaced without tears (the smoke resulting from burning the insides of the Raspberry Pi processor).

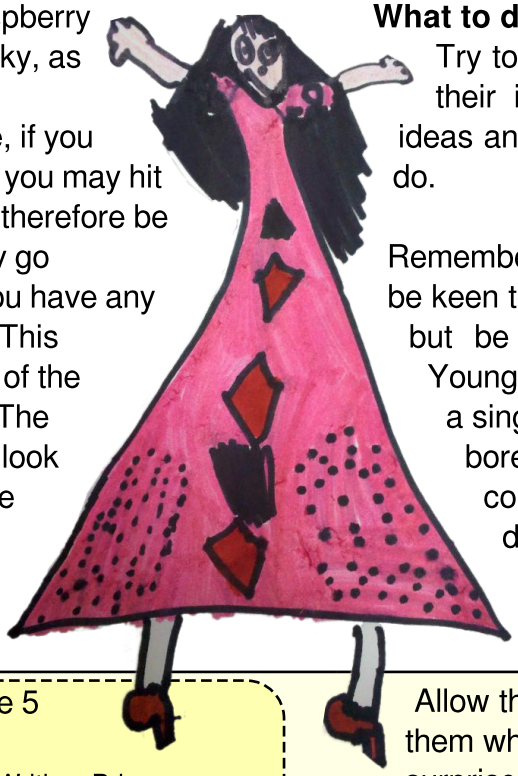
Article by Mike G8NXD
with Meltwater



Meeting Pi

At age 5 or 6 I was fortunate to have access to a ZX Spectrum+ 48K... a passion for electronics and resulting career was the end result. Many years later, the Raspberry Pi is waiting to inspire the next generation of engineers.

When introducing the Raspberry Pi to children it can be tricky, as it is not a polished smooth process yet. Chances are, if you are unfamiliar with things, you may hit the odd roadblock. It may therefore be pertinent to have a sneaky go beforehand and ensure you have any "wrinkles" smoothed out. This includes having a test run of the software you want to try. The key here is making things look simple, as it will encourage them to have the confidence to try it for themselves.



What to do?

Try to tailor the experience to pick up on their interests and skills. Ask them for ideas and suggest possible things they can do.

Remember to take things slowly. You may be keen to show them everything all at once, but be careful not to overwhelm them. Young children may not want to focus on a single thing for long periods. If they get bored, switch to something else or come back to it another day - let them drive it. Pitch things at their level and remember to let go of the mouse and keyboard!

Test Subject - Rosie Age 5

Interests:

Animals, Drawing, Reading, Writing, Princesses, Anything Pink, Singing and Dancing.

1. Hardware Introductions

Prior to getting my Raspberry Pi, I introduced the idea of circuits, electronics and computers. First, I opened up a desktop PC and allowed some hands on (tightening of screws while I replaced the power supply) and explained the various parts and what they did (in a general basic level), as well as some basic electrical safety!

I'd previously purchased a $5 TI LaunchPad development board since it was ideal for pre-RPi playing. I used this to familiarise her with bare component boards, understanding the idea of touching and interacting with switches and lights. Alternatively any basic child / baby toy could be dismantled to give a similar introduction (power, input, control, output etc)

Just keep a balance, don't get carried away and flood them with information.

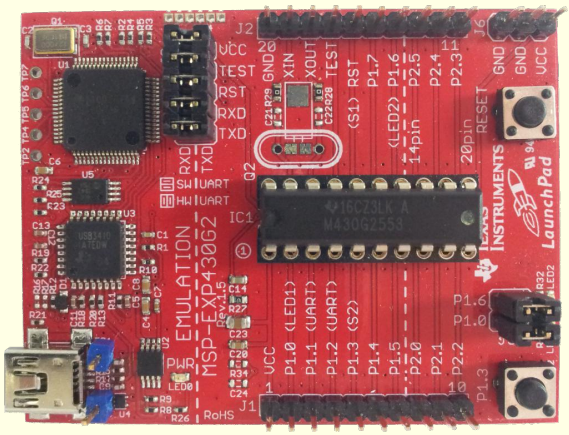
Allow them to ask about any parts and ask them what they think things do (you may be surprised at some of the answers you get!).

Impressed rating: 4/5

Seeing the guts of a real computer is exciting and making things happen by pressing buttons gives a sense of control.

Independence rating: 0/5

You will need to take the lead in this activity to direct and teach them about simple electronics.



TI LaunchPad (http://www.ti.com/launchpad)

2. Drawing

You can introduce control of the mouse and co-ordination simply by using a basic drawing application. Starting with a simple brush you can then introduce shapes and fill tools, which will allow them to generate some interesting and colourful drawings. This is excellent to build up their confidence if they are not familiar at all with computers. You can try GNU Paint (gpaint), which is very similar to MS Paint in Windows.

Impressed rating: 2/5

She has done it on other computers.

Independence rating: 4/5

This was easy to pick up without much assistance.

3. Information Super-Highway

To help with her project on zebras, I showed her how to Google some facts and pictures of zebras using her Raspberry Pi.

Impressed rating: 2/5

Again we have done this on other computers before.

Independence rating: 3/5

Ensure you enable "Safe Search" and guide them to useful known sites (BBC, Wikipedia etc.)

4. Teaching "Scratch" To Draw Spirals

By using some basic steps in Scratch it was easy to instruct him (the cat) to dart around making spirals. After the basic program was set up I asked Rosie what she wanted him to do? Smaller spirals? Larger? The other direction? She replied, "Colours"... so we made one of the spirals green and the other blue.

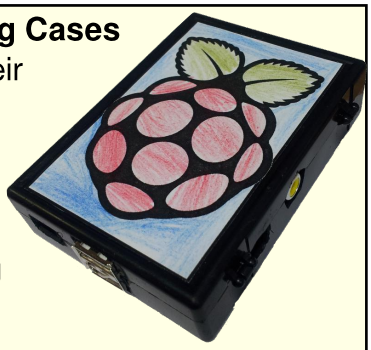
Now leaving her to run it, and showing her how to get back to the main screen each time, she was controlling it herself and making the changes. I turned away for a few minutes and magically Scratch was drawing in pink!

A few more adjustments and Scratch was spinning out several spirals in multicolours in all different directions. Not bad going for a first visit!

Impressed rating: 4/5 Even simple spirals were lots of fun, simply because she could control them.

Independence rating: 4/5 With time I think she would pick up most of the concepts. Even at age 5 she was able to read and understand the different Scratch commands.

5. Making/Decorating Cases



Creating a case for their Raspberry Pi makes a great project, be it made with lego, cardboard, or decorating an exisiting box.

Impressed rating: 4/5

Adding their own touches will make it theirs.

Independence rating: 5/5

Let them do as much or as little as they like.

6. Writing...

It may seem totally silly to most of us, but even a basic text editor (leafpad) with no features at all can inspire a child to create and learn. By encouraging them to write their own stories, or even copy one of their favourites, isn't just a good English lesson but it helps them master the archaic layout of the keyboard and get comfortable with the computer.

Add a simple word processor (e.g. Abiword), which supports images. You can create mini-story books with images using the internet (search for: "cartoon castle", "clipart princess" etc... obviously due to copyright any images will need to be for personal use).

Impressed rating: 4/5

Writing real stories is lots of fun (and yes...if they can copy a story, then they can copy programming code, in time...).

Independence rating: 3/5

This will depend on their writing and keyboard skills to determine how much or little help they will need.

So far I've only just covered some of the basics the Raspberry Pi is able to do, even for a child as young as 5. The key is to give them the tools, show them how and let them experiment. Step away from the keyboard and let them wobble the mouse cursor around. Let them try, they'll ask for help if they need it.

Article by Meltwater
(Drawings by Rosie)

C is an excellent language for building fast and very efficient programs. It can be found in applications where speed and memory footprint matter, such as the Linux kernel and data acquisition systems.

When learning a new language it is a good idea to first solve the programming problem with pen and paper before reaching for the keyboard. Pseudo-code and flow diagrams provide methods for doing this. Particular languages tend to prefer particular solutions to a given problem, however this is something that can be understood with practice.

1 – Getting off the ground

Right, time to get something running. Here is a first C program,

```
#include <stdio.h>
int main()
{
  /* Print a string to the screen. */
  printf("In the beginning...\n");
  /* Return 0 to signal success. */
  return 0;
}                                        first.c
```

Edit, compile and run

The `first.c` program should be typed in using a text editor. For this tutorial `nano` is used as an example text editor. Some useful `nano` commands are given in the table below,

Command	Meaning
`nano file.c`	Open a new or existing file called `file.c`
Ctrl+O	Write the current file to disk
Ctrl+X	Exit and ask if the current file should be written
Ctrl+W Ctrl+T 10	Navigate to line number 10

Open two terminal windows, one for `nano` and the other for compilation. (Using the default Raspberry Pi Debian image, a terminal window can be opened by clicking on `Accessories` and then `LXTerminal`.)

Use `nano` to create a new file called `first.c`. Type in the program and save it. Before a C program can be run it has to be compiled. In Linux the GNU C compiler (gcc) is used to compile C source code. In the other terminal window, compile the program by typing:

```
gcc -o first first.c
```

The gcc compiler will then attempt to compile the C program and link it together with the standard libraries to form an executable called first. If the compiler is successful, no messages will be printed on the screen. If an error is reported by the compiler, check the line number where the error is reported and try to compile the program again. When the program has been successfully compiled, run the executable by typing:

```
./first
```

This will print the string onto the screen and return zero to the operating system. In the BASH shell the output value from this return statement is stored in the `$?` variable. This variable holds the return statement from the previous command only. Type `echo $?` to print the value of `$?` on the screen.

Printing a string

The execution of the `first.c` program starts from the `main()` function. The function has an integer (`int`) return value, which is given to the left of the function name. Any arguments passed into the function would go within the parentheses `()`. In this case, no arguments are passed into the `main` function. The body of the function is defined by the brackets `{}`, which is referred to as a compound statement. A semicolon is used to complete each line or statement within the compound statement. In the first program there are only two statements, (i) a `printf` function call to print a string to the screen and (ii) the return statement. The `printf` function is a standard library function, but for the program to be compiled its declaration has to be included by including `stdio.h`. The linker then finds the library that contains the implementation of this function and forms an executable.

When the `printf` statement is called it is given a string. The string is delimited by quotation marks and contains the new line character `\n`. Before the `printf` function call, a comment is written between the `/* */` braces. When writing programs, it is a good idea to add comments to explain the program to other users or for long-term documentation.

Keyboard and screen

There are many standard library functions defined in the `stdio.h` header file. These functions can be used to read from the keyboard and write to the screen. Input values can be read from the keyboard using the `scanf` function,

```
#include <stdio.h>
int main()
{
  /*Declare an variable to hold a number */
  int age;
  /* Ask for input */
  printf("How old are you? ");
  /* Read a number from the keyboard */
  scanf("%d",&age);
  /* Echo the age */
  printf("You are %d years old.\n",age);
  return 0
}                                          age.c
```

In the `age.c` program an integer variable is declared to hold a value. The integer variable corresponds to a space in the memory of the computer, which is allocated to hold the value. The user is asked their age. Then the program waits for input from the keyboard. The input is read from the keyboard using the `scanf` function. The character code `%d` tells `scanf` to read an integer from the keyboard. The integer is written into the memory allocated to the `age` variable by passing the memory address of the `age` variable, which is accessed via `&age`. The age value is then written back to the screen using `printf`. Notice that `printf` also uses the `%d` character code to denote integer output. More information on functions in `stdio.h` can be found by typing:

```
man stdio
```

To exit the manual page, press `q`. There are many Linux manual pages which cover other standard library functions and the compiler options (`man gcc`).

Simple mathematical operations

Computers are able to perform many mathematic calculations quickly. For example,

```
#include <stdio.h>
int main()
{
  /* Declare three integer variables */
  int x, y, z;
  printf("enter two whole numbers, separated by
a space: ");
  /* Read two values from the keyboard */
  scanf("%d %d",&x,&y);
  /* add the two values together and place the
result in z */
  z = x + y;
  printf("%d + %d = %d\n",x,y,z);
  /*multiply x by 10 and then assign the result
to x */
  x = x * 10;
  printf("%d + %d = %d\n",x,y,x+y);
  return 0;
}                                  simple_math.c
```

The program `simple_math.c` demonstrates addition (`+`) and multiplication (`*`) operators. In a similar manner, subtraction (`-`) and division (`/`) can also be used. Mathematical expressions within a program are executed sequentially. For example, `x=x*10` will start from the current value of `x`, multiply it by ten and assign the result to `x`.

Challenge question

Find the errors in the program given below. There are four errors, which need to be corrected before the program will compile successfully.

```
int main
{
  int i = 100, j = 9, k;
  i = i/10;
  k = i - j
  printf("Well done this program
compiles.\n");
  print("%d - %d = %d\n",i,j,k);
  return 0;
}
```

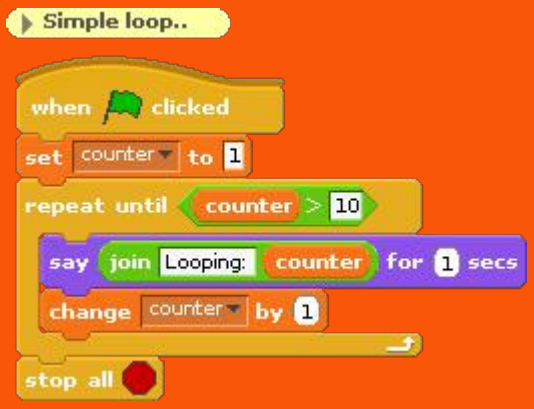
The solution will be given in the next tutorial.

Article by W. H. Bell & D. Shepley

THE SCRATCH PATCH

In this article, I'm going to show you a few programs to try in Scratch.

Have a play around with them until you are familiar with how to use variables, conditionals and loops. With these basic programming tools, there's a huge amount you can do!



Hello User!
when clicked
ask What's your name? and wait
set name to answer
say join Hello name for 2 secs
stop all
This would work too. But it's clearer if you use a variable.
say join Hello answer for 2 secs

The Scratch Forums

If you like Scratch, I'd recommend checking out the forum. It's a very friendly online community.

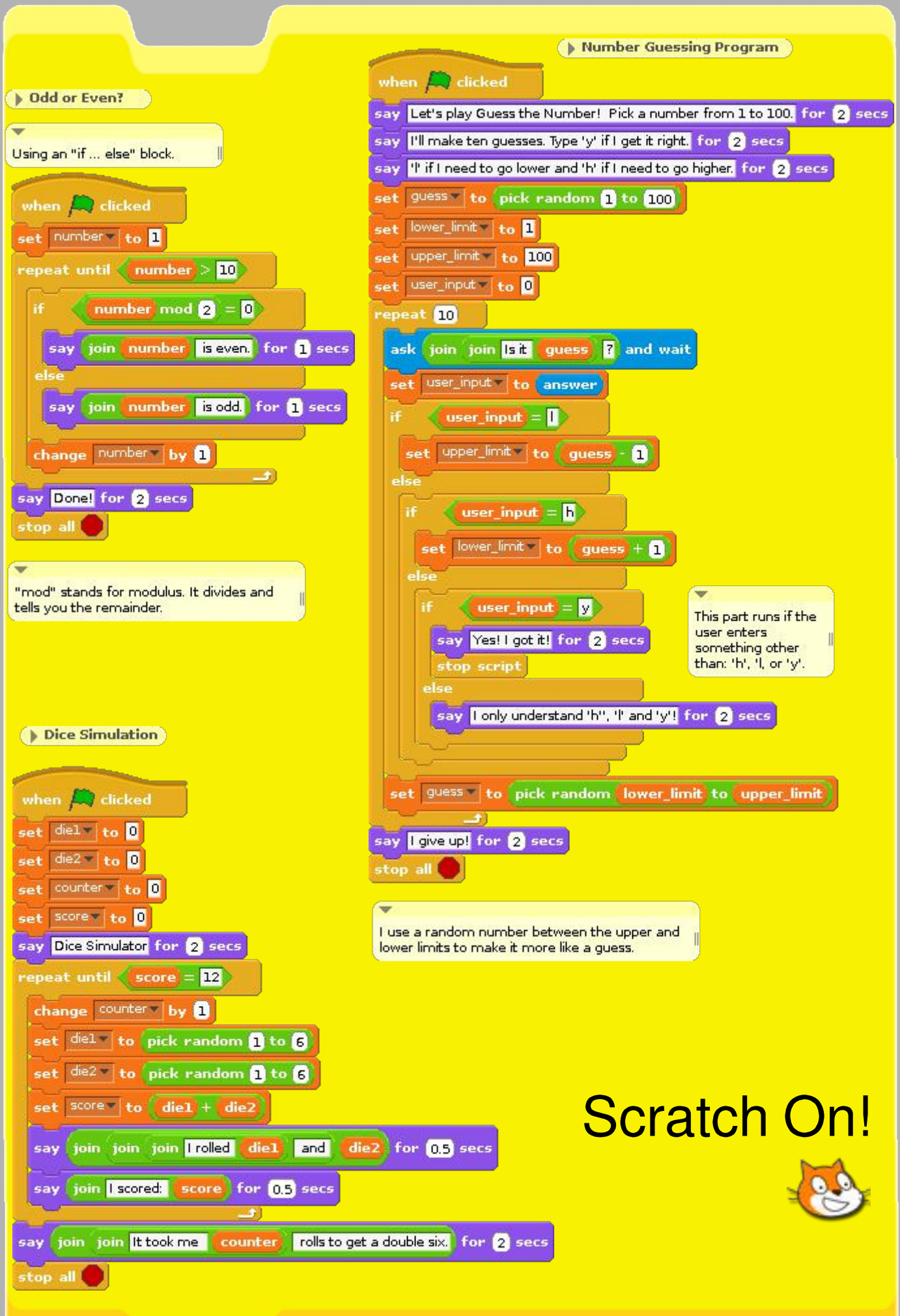
You can upload your projects and share ideas with other Scratchers!

http://scratch.mit.edu/forums

This article is by antiloquax.

My Scratch id is 'racypy' and you can find all the scripts featured here in my projects.

If you have any questions, requests or suggestions, please email me:
antiloquax@sky.com


Odd or Even?
Using an "if ... else" block.
when clicked
set number to 1
repeat until number > 10
if number mod 2 = 0
say join number is even. for 1 secs
else
say join number is odd. for 1 secs
change number by 1
say Done! for 2 secs
stop all
"mod" stands for modulus. It divides and tells you the remainder.
Dice Simulation
when clicked
set die1 to 0
set die2 to 0
set counter to 0
set score to 0
say Dice Simulator for 2 secs
repeat until score = 12
change counter by 1
set die1 to pick random 1 to 6
set die2 to pick random 1 to 6
set score to die1 + die2
say join join join I rolled die1 and die2 for 0.5 secs
say join I scored: score for 0.5 secs
say join join It took me counter rolls to get a double six. for 2 secs
stop all
Number Guessing Program
when clicked
say Let's play Guess the Number! Pick a number from 1 to 100. for 2 secs
say I'll make ten guesses. Type 'y' if I get it right. for 2 secs
say 'l' if I need to go lower and 'h' if I need to go higher. for 2 secs
set guess to pick random 1 to 100
set lower_limit to 1
set upper_limit to 100
set user_input to 0
repeat 10
ask join join Is it guess ? and wait
set user_input to answer
if user_input = l
set upper_limit to guess - 1
else
if user_input = h
set lower_limit to guess + 1
else
if user_input = y
say Yes! I got it! for 2 secs
stop script
else
say I only understand 'h', 'l' and 'y'! for 2 secs
set guess to pick random lower_limit to upper_limit
say I give up! for 2 secs
stop all
This part runs if the user enters something other than: 'h', 'l, or 'y'.
I use a random number between the upper and lower limits to make it more like a guess.
Scratch On!

the python pit™

In Issue 2 we showed how you can create a graphics screen surface and draw shapes onto it.

This month we will look at overlaying more surfaces on top of the screen surface. You can think of each surface as being a separate window within the main screen surface. Don't worry if this sounds complicated, the following examples should help to demonstrate the principle.

NOTE: For these examples, you will need both Python and Pygame installed on your computer.

We will draw a simple picture with some grass, sky and a sun. We could draw this all on one surface. The advantages of giving each object its own surface come when you wish to move objects around. If you drew the sun onto the same surface as the sky and then attempted to move the sun, you would leave a yellow trail where the sun used to be. To fix that you would have to redraw the sky every time the sun moved. With layered surfaces you don't have to worry about your graphics leaving trails on the screen like an etch-a-sketch.

```
# THREE SURFACES

# By Jaseman - 13th June 2012

import os, pygame; from pygame.locals import *
pygame.init(); clock = pygame.time.Clock()
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Three Surfaces")
screen = pygame.display.set_mode([400,200],0,32) # The main screen
sky = pygame.Surface((400,200)) # A sky surface
sky.fill((200,255,255)) # Fill the surface in light blue color
grass = pygame.Surface((400,100)) # A grass surface
grass.fill((50,150,50)) # Fill the surface in green color
sun = pyagme.Surface((40,40)) # A sun surface
pygame.draw.circle(sun,(255,255,0),(20,20),20)
screen.blit(sky,(0,0)) # Paste the sky surface at x,y
screen.blit(sun,(180,30)) # Paste the sun surface at x,y
screen.blit(grass,(0,100)) # Paste the grass surface at x,y

pygame.display.update()
pygame.time.wait(10000) # A 10 second pause before ending the program
```

You will notice that the sun surface has a black background color. Surfaces are always rectangular.

PYTHON VERSION: 2.6.6 / 3.2.2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 6 / Win7

TESTED!

We can make the sun surface appear to be round rather than rectangular by making black into a transparent color. When it is transparent the blue of the sky will come through, filling those black corners...

We make black a transparent color on the sun surface by adding the line:
`sun.set_colorkey([0,0,0])`

This means anything that is drawn on the sun surface rectangle in color[0,0,0] will be transparent. The graphics on surfaces underneath will show through the transparent areas.

This time we also alter the x,y position onto which the sun surface will be pasted. We will use the mouse coordinates as the location for the sun and have the program run in an endless loop.

```
# THREE SURFACES V2

# By Jaseman - 13th June 2012

import os, pygame; from pygame.locals import *
pygame.init(); clock = pygame.time.Clock()
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Three Surfaces")
screen = pygame.display.set_mode([400,200],0,32) # The main screen
sky = pygame.Surface((400,200)) # A sky surface
sky.fill((200,255,255)) # Fill the surface in light blue color
grass = pygame.Surface((400,100)) # A grass surface
grass.fill((50,150,50)) # Fill the surface in green color
sun = pyagme.Surface((40,40)) # A sun surface
sun.set_colorkey([0,0,0])
pygame.draw.circle(sun,(255,255,0),(20,20),20)

while True: # A never ending loop to keep the program running
      for event in pygame.event.get():
            if event.type == QUIT:
                  pygame.quit()
                  sys.exit()

      mousex,mousey = pygame.mouse.get_pos()
      screen.blit(sky,(0,0)) # Paste the sky surface at x,y
      screen.blit(sun,(mousex,mousey)) # Paste the sun surface at x,y
      screen.blit(grass,(0,100)) # Paste the grass surface at x,y

      pygame.display.update()
```

When you run the program and move the mouse around, notice how the sun behaves. If you move the sun below the level of the grass it goes under the grass surface. This is to do with the order in which the surfaces are pasted onto the screen. If you were to switch the `screen.blit` commands around for the sun and grass, then the sun will be above the grass surface.

Notice the `while True` section has an event handler added in to deal with the QUIT event. This just helps to shut down the program smoothly when you tell it to quit.

PYTHON VERSION: 2.6.6 / 3.2.2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 6 / Win7

TESTED!

the python pit

Continued over page...

In this slight variation we've made the sun bigger, put a black circle in the centre of it and introduced a new command to give the entire sun surface a 50% transparency:
`sun.set_alpha(128)`

The value 128 can be any number between 0 and 255; 255 being solid and 0 completely transparent.

```
# THREE SURFACES V2

# By Jaseman - 13th June 2012

import os, pygame; from pygame.locals import *
pygame.init(); clock = pygame.time.Clock()
os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("Three Surfaces")
screen = pygame.display.set_mode([400,200],0,32) # The main screen
sky = pygame.Surface((400,200)) # A sky surface
sky.fill((200,255,255)) # Fill the surface in light blue color
grass = pygame.Surface((400,100)) # A grass surface
grass.fill((50,150,50)) # Fill the surface in green color
sun = pyagme.Surface((80,80)) # A sun surface
sun.set_colorkey([0,0,0])
sun.set_alpha(128)
pygame.draw.circle(sun,(255,255,0),(40,40),40)
pygame.draw.circle(sun,(0,0,0),(40,40),15)

while True: # A never ending loop to keep the program running
      for event in pygame.event.get():
            if event.type == QUIT:
                  pygame.quit()
                  sys.exit()

      mousex,mousey = pygame.mouse.get_pos()
      screen.blit(sky,(0,0)) # Paste the sky surface at x,y
      screen.blit(grass,(0,100)) # Paste the grass surface at x,y
      screen.blit(sun,(mousex,mousey)) # Paste the sun surface at x,y

      pygame.display.update()
```

Because the inner circle is coloured black, it becomes transparent, giving the effect that the surface has a hole in it. Move the mouse across the horizon and watch carefully what happens.

PYTHON VERSION: 2.6.6 / 3.2.2
PYGAME VERSION: 1.9.2a0
O.S.: Debian 6 / Win7
TESTED!

We would recommend you to do some more experimenting with the techniques demonstrated here. Try making your own surfaces, drawing shapes onto them in different colours, and using the `surface.alpha()` and `surface.set_colorkey([])` commands to create transparency effects. Mastering these will be essential if you are considering writing games, but they may also be useful in other applications where display effects are required.

We enjoy getting your feedback, so please let us know how you get along.

the python pit

> PROGRAMMING FUNDAMENTALS
> BY ALEX KERR

Introduction

Although there are many programming languages, a lot of it is just different ways to do the same thing; much like spoken languages all have different ways of saying “Hello, how are you?”. They all mean the same thing once translated, but they’re said differently. This section aims to explain some of the basic fundamentals of programming that are useful to know for every language.

Variables

Variables are pieces of data that can change throughout the course of the program. There are different types of variable and they store different kinds of data. Integers, for example, store whole numbers while strings store text. A user input, for example, would be a variable. Without variables, programs would do the same thing every time, which isn’t very useful.

'If' Statements

‘If’ statements allow us to check things within the programs. If the condition being checked is true, then the code within it executes. If not, then that portion of the code is skipped. This allows us to act upon variables. For example, we could check if the player of a game has scored enough points to continue.

We can expand on ‘if’ statements by telling the program to do something if the condition is met and something else if the condition is not met.

We can also expand upon ‘if’ statements more by checking two or more conditions, such as if a character is touching a danger AND isn’t protected. We call these logical operators.

The three are AND, which returns true when both tests are true, OR, which returns true if one or more of the test conditions are true and NOT, which returns true if the input is false, and vice versa.

Loops

Loops repeat a piece of code until a certain condition is met. This could be used to repeat an animation, for example.

There are different kinds of loops. There are ‘do until’ loops, which repeat the code while the test condition is false, and ‘do while’ loops, which executes the code while the test condition is true.

With this knowledge, learning to program in any language should be easier. Try a few examples in Scratch using the blocks shown to see them in action.

Article by Alex Kerr

Feedback

'The MagPi kind of reminds me of the magazine that I used to get as a kid to help me out with my Acorn Electron...10 REM and so on!'

Simon Frost

'Your articles and easy to follow guides are an absolute lifesaver.'

David Deen

'Excellent publication and extremely informative. Reading this really did take me back to the 80's when computing was about the fun of learning. Back then the magazines got me into learning BASIC and now with your excellent articles I think I will be starting out in Python. This is really inspiring and just what computing needs to get people interested again. Well done and please keep it coming!'

Philip Tyerman

'What a fantastic magazine. Just received my Raspberry Pi in the post today and the articles in issue 2 of The MagPi just hit the spot. Brilliant.'

John Franks

'The magazine is awesome! I can now get back to programming, which I never did after BASIC stopped being bundled with computers!'

Francis Medeiros

'I love reading this magazine, it is great. I would love to subscribe to this and get a hard copy delivered to me every month.'

Jack Moorhouse

The forums are ok, but there is a real lack of hand holding and starting from the beginning, which your magazine seems to have got spot on.'

Alex Wilkinson

'Thank you for putting this together. It helps newbies like myself to get a head start.'

Noris

'I've had a quick skim through it and the contents look great. Really nice close up photos of the Pi and lots of detailed descriptions of what does what. Plenty for everyone. That's my evening sorted!'

Grumpyoldgit

'Loving the magazine. I'm new to programming so finding the magazine very useful.'

Richard

'Congratulations. A very good magazine for content. But its heavy use of dense colours - which I do not think adds to the appeal - means that I am very loath to print it. Also such use of colours makes the magazine quite difficult to read for many people with various types of colour blindness.'

Peter

'I would like to take a moment and thank you for the efforts put forth for the magazine! I have really enjoyed both issues.'

Cody

'The MagPi must be the most useful and helpful tech magazine I've ever read... and that's a lot of magazines.'

Patrick Duensing

'I absolutely love the idea of having a monthly magazine about the Raspberry Pi that is free to download and has great tutorials, software and hardware information, with pictures and well written text. I love how things are going so far. I would like to recommend that you guys set up some sort of donation spot so I may donate to you guys. I love everything you are doing so far! Keep up the great work!'

Devin

'I've just downloaded issue 2 and printed it. Great quality and info. Thanks so much for all your hard work.'

Bigsi111

Image from photofunia.com

The MagPi™

Raspberry Pi is a trademark of the Raspberry Pi Foundation. The MagPi magazine is collaboratively produced by an independent group of Raspberry Pi owners, and is not affiliated in any way with the Raspberry Pi Foundation. The MagPi does not accept ownership or responsibility for the content or opinions expressed in any of the articles included in this issue. All articles are checked and tested before the release deadline is met but some faults may remain. The reader is responsible for all consequences, both to software and hardware, following the implementation of any of the advice or code printed. The MagPi does not claim to own any copyright licenses and all content of the articles are submitted with the responsibility lying with that of the article writer.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

HELP WANTED!

The MagPi Team are looking for volunteers to assist with:
page layouts (Scribus), artwork, photography, testing and other administration.

The current team are putting in SERIOUSLY long hours, so please do help out. Working on the magazine, which is read by over 100,000 people, is a great educational experience.

We would expect at least a few hours of your evening or weekend each week.

editor@themagpi.com

Other Resources and Weblinks

http://www.themagpi.com
Official website of The MagPi magazine.

http://www.raspberrypi.org
Official home of the Raspberry Pi Foundation

http://inventwithpython.com/IYOCGwP_book1.pdf
Invent Your Own Games with Python

http://thepythongamebook.com
The Python Game Book (DocuWiki)

http://www.cprogramming.com/
C and C++ Programming Resource

http://www.element14.com/community/groups/raspberry-pi
Element 14 Community Group

The MagPi Issue 03 JUL 2012

Team:

Ash Stone
Chief Editor / Administrator

Jason 'Jaseman' Davies
Editor / Page Designs / Writer / Website

Tim 'Meltwater' Cox
Writer / Editor / Photographer / Page Designs

Chris 'tzj' Stagg
Writer / Editor / Photographer / Page Designs

Bobby 'bredman' Redmond
Writer / Page Designs

Darren Grant
Writer / Page Designs

0The0Judge0
Python Tutor / Administrator

Antiloquax, Alex Kerr, W.H. Bell,
D. Shepley, Mike G8NXD
Writers

Print Edition Preparation:
W.H.Bell, Ian McAlpine, Colin Deady, Aaron Shaw,
Isa McKenty, Simon Johnson,
Alan 'paisleyboy' Holroyd, Sam Marshall,
Joshua Marinacci, Colin Norris, Mark Robson,
Steve Drew, Alex Baker, Adrian Harper

Raspberry Pi is a trademark of the Raspberry Pi Foundation.